AFRL-IF-RS-TR-2001-236
Final Technical Report
October 2001

ITSY HANDHELD SOFTWARE RADIO

Vanu, Inc.

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J213

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2001-236 has been reviewed and is approved for publication.

APPROVED:

SIAMAK S. TABRIZI
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

ITSY HANDHELD SOFTWARE RADIO

Vanu G. Bose

Contractor: Vanu, Inc.
Contract Number: F30602-99-C-0173
Effective Date of Contract: 30 August 1999
Contract Expiration Date: 8 March 2001
Program Code Number: J213
Short Title of Work: ITSY Handheld Software Radio

Period of Work Covered: Aug 99 – Mar 01
Principal Investigator: Vanu G. Bose
Phone: (617) 864-1711
AFRL Project Engineer: Siamak Tabrizi
Phone: (315) 330-4823

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Siamak Tabrizi, AFRL/IFGC, 26 Electronic Pky, Rome, NY. 13441-4514.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 2001	Final Aug 99 - Mar 01

4. TITLE AND SUBTITLE

ITSY HANDHELD SOFTWARE RADIO

5. FUNDING NUMBERS

C - F30602-99-C-0173
PE - 62302E
PR - H866
TA - 00
WU - 01

6. AUTHOR(S)

Vanu G. Bose

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Vanu, Inc.
One Porter Square, Suite 18
Cambridge MA 02140

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFGC
26 Electronic Pky
Rome NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2001-236

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Siamak S. Tabrizi/IFGC/(315) 330-4823

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release: distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Software radio technology promises to eliminate existing radio interoperability problems, provide a path for rapidly incorporating advances in digital communications, and enable new applications and waveforms that take advantage of the tremendous flexibility. However, the current state of low power processing technology limits application to low power handheld radios. A handheld software radio platform would enable the construction of devices that could inter-operate with multiple legacy systems, download new waveforms and be used to construct adhoc networks that optimize the physical layer for the current conditions. Low power operation is the most significant challenge facing software radio. While some approaches sacrifice considerable flexibility by using technologies such as re-configurable logic, we believe that the advances in processor power management will make a pure software radio approach feasible in a handset in 3-5 years. A prototype handheld software radio based on the 200 MHz StrongARM processor was designed and fabricated under this contract, and several representative waveforms were ported to the platform. This work enabled us to evaluate the capabilities handheld software radio based on existing processor technology, and provides benchmarks that can be used to assess the capabilities of software radio systems based on future lowpower processors.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Radio, Handheld Software Radio	28
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Table of Figures

Acknowledgements

Vanu, Inc. would like to acknowledge the support provided by Compaq Computer Corporation, and specifically Carl Waldspurger, Bill Hamburgen and Debbie Wallach of Compaq research. Compaq loaned several ITSY pocket computers for this project, and considerable assistance was provided by members of the Compaq research team in getting Linux running on the ITSY and providing tools and support for debugging.

Summary

Software Radio is an emerging technology that enables a single piece of hardware to communicate with a wide range of wireless system simply by running a different program. While software radio holds considerable promise, the processing power required limits battery life when using today's low power processors. The purpose of this work was to evaluate the capabilities of handheld software radio using existing processors.

In order to perform the desired benchmarks, we built a handheld software radio platform utilizing the best available components at the time. The benchmarking process consisted of three steps. First we ported the waveforms to a fixed-point implementation, which was necessary since the StrongARM does not have a floating-point unit. Then we benchmarked the waveforms on a StrongARM evaluation platform to get an estimate of the CPU cycles required, and also on Pentium and PowerPC platforms for comparison. Finally, a subset of the waveforms were actually run and measured on the prototype platform.

This project successfully demonstrated a working handheld software radio. This prototype has served as an excellent proof of concept, and has identified the capabilities and limitations of handheld software radio using existing low power processors. The benchmarking results suggest that implementation of many legacy narrowband systems entirely in software using existing low power processors is feasible today. These results can be used to project the future capabilities of software radio given the expected improvements in low power processors.

The major contributions of this work are the demonstration of the feasibility of all-software processing on a general purpose processor for several different legacy waveforms, and identification of the key areas of research and development required to move software radio technology into a viable commercial realm (discussed in the recommendations section at the end of this report), and the measurement of processor performance for five different waveforms that provide a guide for the performance of software radio applications on future processors.

Introduction

Software Radio is an emerging technology that enables a single piece of hardware to communicate with a wide range of wireless system simply by running a different program. A handheld software radio would enable a user to simply tap the appropriate icon and turn their PDA into a cell phone, pager, land-mobile radio, wireless data connection, garage door opener, baby monitor, cordless phone or RF remote control. In addition, many new services or waveforms can be deployed as software upgrades to existing platforms. Cost and size constraints would make it infeasible to consider building all of these functions into a PDA using a traditional hardware radio approach, but software radio permits the construction of a single piece of hardware that is capable of performing all of these functions.

While software radio holds considerable promise, the processing power required limits battery life when using today's low power processors. The purpose of this work was to evaluate the capabilities of handheld software radio using existing processors. Software radio is a systems problem, that involves integrating components from multiple different disciplines, including radio frequency engineering, computer system architecture, operating system design and digital signal processing. As a result, it is difficult to predict where the system performance bottlenecks will arise. For different applications the bottleneck might be in processing, memory bandwidth, operating system overhead or A/D resolution. For this reason, we have chosen the approach of building a working system and running benchmarks on that, rather than running the benchmarks in a simulated environment that does not capture all of the subtleties involved with system integration.

Background

Vanu, Inc. is a pioneer in the software radio industry. The company was founded in 1998 and based on several years of software radio research from the MIT SpectrumWare project [BIWG99]. Vanu, Inc. provides software radio system design consulting services and licenses OEM software radio applications (e.g. GSM, CDPD, cordless phone). Vanu, Inc. has received one of four initial JTRS Step 2B contracts for the development of a military software radio, and is under contract to develop a prototype for a Tier 1 supplier to the automotive industry.

The Vanu, Inc. software radio architecture is illustrated in Figure 1. The architecture consists of three hardware layers, an operating system layer and two stacks of application level components. Each hardware layer has a control interface that allows the application to control and monitor hardware components through device drivers in the operating system. The hardware layers, operating system and device drivers comprise the software radio platform for which Vanu, Inc. supplies design consulting services. Vanu, Inc. licenses software for the upper layers, which specify the waveform, signal processing data path, and control the radio.

The bottom layer represents the antenna system. For receive, the input to this layer is the electromagnetic wave that impinges on the antenna, and the output is an analog RF signal. For transmit the input is an analog RF signal and the output is an electromagnetic wave radiated from the antenna. Software radio systems may require many different types of antennas including antenna arrays and smart antennas that can be controlled to adapt to different types of interference or transmission requirements. The antenna may have multiple outputs, or the output may be selectable between different inputs as in simple diversity systems. The number and type of interfaces can be queried and configured through the control interface illustrated on the right of the antenna layer.

The RF-to-digital layer is responsible for converting the RF signal from the antenna to a digital signal. The interface to the motherboard are two parallel digital data streams for transmit and receive data, as well as clock signals for timing of both transmit and receive. The control interface allows the system to set parameters such as RF center frequency and IF bandwidth.

The motherboard is the digital processing, storage and I/O system, much like the motherboard in a PC. This layer processes the data, and has several data interfaces: the RF-to-digital layer, operating system, network and user I/O interfaces. Data can be read from any of these sources, processed and written to any of the sources. Treating the network interface as a first class data source and sink permits the system to be easily scaled to utilize multiple processors connected to the network for infrastructure applications.

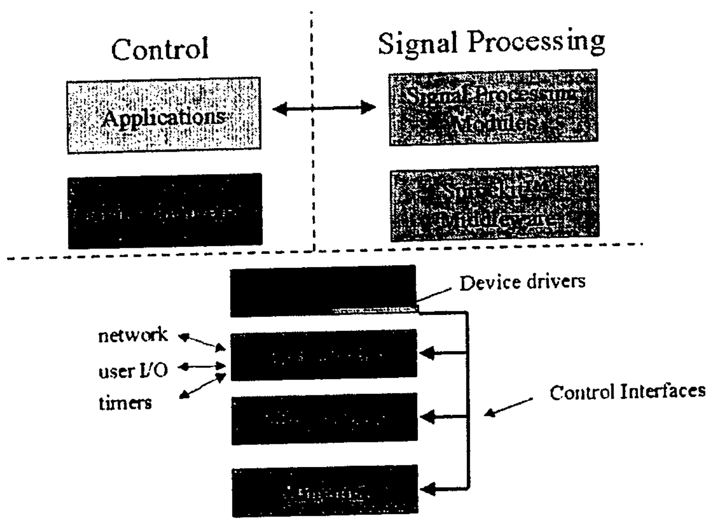

Figure 1: Vanu, Inc. Software Radio Archtiecture

The Operating System layer is the critical abstraction that enables software and hardware to be independently produced. We have chosen Linux as the operating system for several reasons. It is open source and we have the ability to extend it to support the high data rate I/O and other features required by software radios. In addition, the open nature of the operating system prevents a single company using dominance in the operating system market to control the software radio application market. The operating system isolates the hardware from the application level software. Our systems currently run Linux, but the software should be compatible with any reasonably POSIX compliant OS such as VxWorks or QNX. A key aspect of the architecture is that all data and control flow across the operating system boundary, and we have engineered mechanisms to efficiently implement these information flows. This enables the data path as well as the system parameters to be defined by software.

The software that sits on top of the operating system is divided into two functional groups: control and signal processing. The signal processing software stack consists of the Sprockit™ signal processing middleware and the signal processing library modules. Sprockit™ provides a modular object oriented framework for constructing software radio applications. The signal processing modules are a set of signal processing functions for constructing radios, such as channel selectors, demodulators and decoders. These modules perform the signal processing, while Sprockit™ handles the I/O between these modules.

The control software stack consists of the software radio applications and the Radio Description Language (RDL) interpreter [Chapin01]. The software radio applications completely specify both the data path and the configuration for a radio application. These applications are written in portable, interpreted code that is processed by the RDL interpreter. The interpreter sets up the appropriate modules in the specified data path layout and configures the parameters on each module. Once the application is running, a module can trigger signals that pass control back to the RDL code, which can take appropriate actions, such as sending control signals back to that module or other modules. These signals could be used to indicate events such as reception of a complete packet, or conditions such as low received signal strength.

This architecture supports a wide range of hardware implementation technologies in each layer. This flexibility enables significant use of existing technology, as well as future technology tracking. The architecture also supports a wide variety of target platforms. Single processor handheld devices and multiple processor fixed-wireless infrastructure are both well supported by this architecture.

Report Outline

The remainder of this report begins with a description of the system design and development as well as the methodology for measuring waveform performance. The Results and Discussion section describes the finished prototype and the results of the waveform benchmarking experiments. These benchmarks are then discussed in the context of future low power processor improvements. The conclusion summarizes the work and the key results, and is followed by the Recommendations section, which provides recommendations for future research given the results of this work.

Methods, Assumptions and Procedures

In order to perform the desired benchmarks, we built a handheld software radio platform utilizing the best available components at the time. This section describes the design of the handheld platform, and the experimental methodology used to produce the waveform benchmark numbers.

Platform Design

Our design was based on a four layer stack, in order to preserve design and testing flexibility for each portion of the system. The four layers are the processor and user interface, digital I/O card, converter card and the RF front-end. The design of each of these layers and the integration of the system are described in the following sections.

Processor and User Interface

We chose to base the handheld software radio processing and user interface system around the ITSY PDA prototype from Compaq research labs, which is pictured in Figure 2. The ITSY is a PDA built around a 200 MHz StrongARM processor with 16 MB of DRAM and 4 MB of Flash memory. It has a 320x200 gray scale display, 10 buttons, and IrDA port, a serial port and a connector to the memory bus. The memory bus connector was the primary reason for choosing the ITSY, since it provided a high bandwidth path to stream samples directly to and from memory. We were able to obtain several ITSYs on loan from Compaq's western research laboratory. Since this project was started, several commercial PDAs have appeared on the market with suitable expansion bus slots, including Compaq's own iPAQ PDA.

The ITSY processor is responsible for processing the digitized radio data and supporting the user interface that consists of a display, buttons, speaker and a microphone. The Vanu, Inc. software, including Sprockit™ and the signal processing libraries run on the StrongARM resident on the ITSY. Another design choice might have been to use separate processors for the user interface and the software radio processing, which has the advantage that the user functions will not be impacted by the processing load presented by the software radio functions. However, if a single processor can handle both functions, this would be a significant cost savings for the system. The utility of having all processing performed on a single processor was one of the evaluation criteria for the ITSY software radio prototype.

Figure 2: ITSY PDA

Digital I/O Card

The digital I/O card, pictured in Figure 3 is responsible for transferring received digitized data from the A/D converter to the memory bus, and for transferring transmit data from the memory bus to the D/A converter. The board consists of an FPGA for handling the signaling between the memory bus and the converters, 64K dual ported SRAM for buffering data for the memory bus, and an Analog Device 6620 digital downconverter. 32K of the SRAM was used as a transmit buffer and 32K for a receive buffer. The digital downconverter performs a programmable filtering and sub-sampling operation. This chip was included in the system design because early benchmarks showed that up to 40% of the processing power for a given waveform was consumed by this function. By incorporating this programmable chip, we were able to offload a considerable portion of processing from the StrongARM and support a wider range of waveforms. The white connector on the board mates to the ITSY memory bus connector, and the connection to the converter board is on the bottom. In an actual product, this functionality would not be on a separate board. However, for a research prototype, this provided the flexibility to experiment with different components if necessary.

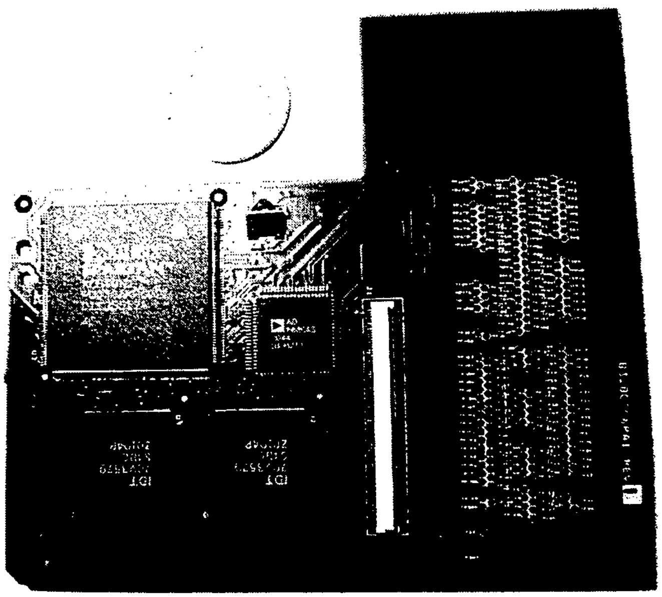

Figure 3: Digital I/O Card

Converter Board

The converter board, pictured in Figure 4, supports the analog-to-digital (A/D) and digital-to-analog (D/A) conversion functions.

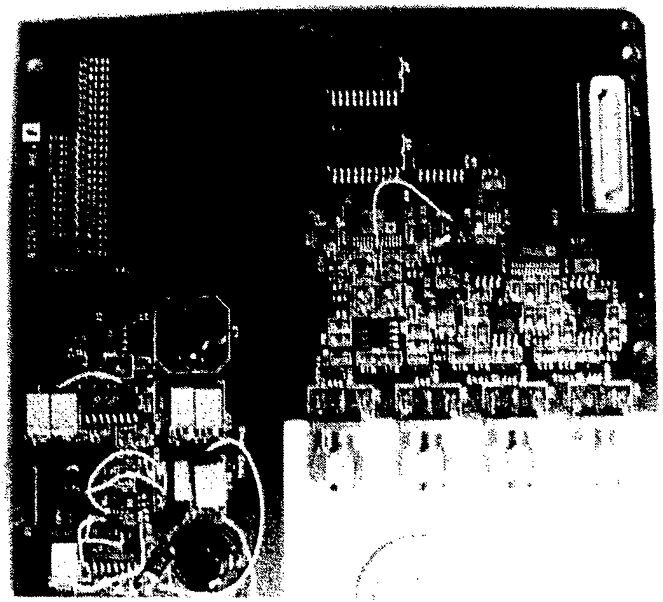

Figure 4: Converter Board

This board has two separate, selectable, analog interfaces, baseband I/Q signals accessible through the SMA connectors at the bottom of the board, and an interface to the Racal MBITR front-end through the white connector at the top right of the board. Multiple interfaces to the RF front-end were built in to permit experimentation with other front ends in the future. The A/D converter is the Analog Devices 9201 and the D/A converter is the Analog Devices 9761. The board supports and AGC control line as well as a received signal strength indicator. The board takes a single voltage supply input in the range of 7-12 volts, and produces voltages of 3.3V, 5V and 6V to power the logic, interfaces and RF front-end

RF Front-end

The RF front-end was purchased as a separate component. Our intial plans were to use the Rockwell Collins Glomo Radio developed under the Darpa Glomo program. This radio covered the frequency range from 2 MHz to 2 GHz, and provided a digital output that consisted of 10 MHz of the spectrum. We obtained a Glomo Radio from Rockwell, and while we were able to properly receive signals from it, we were unable to get it to transmit properly. We then turned to the MBITR front-end, which we obtained from Racal (now Thales) Communications. This front-end covered the range of 30 MHz – 512 MHz and was limited to a 25 kHz channel. While this front-end was limited in functionality compared to the Rockwell, we were able to obtain he necessary tools and support from Racal to integrate the front-end into our system.

Prototype System Integration

The four boards were integrated into a working prototype, shown in Figure 5. The MBITR front-end interfaced to the converter board through the MBITR connector on the daughter board. This connector supported the analog transmit and receive signals as well as serial control for setting the center frequency and adjusting the AGC and a line for monitoring the received signal strength. The control signals emanate from the ITSY, and are passed through the FPGA on the I/O board to the converter board.

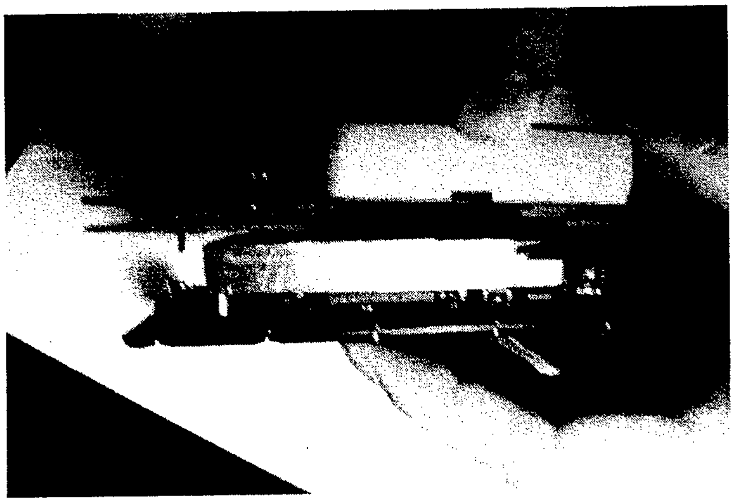

Figure 5: Integrated Handheld Software Radio System

Several of the ITSY buttons were re-mapped to provide a user interface. One button provided an interface to scroll through available *radios*, while a second served as a push-to-talk button for half-duplex systems. The user would toggle the select button until the appropriate radio was chosen. At this point the ITSY is running the chosen radio program and listening to the appropriate channel for a signal. When a signal is detected, the ITSY demodulates the signal, activates the speaker and plays out the audio. If the user depresses the push to talk button, then the ITSY activates the microphone, modulates the signal from the microphone and sends the modulated data to the transmitter.

Waveform Benchmarking Procedure

The benchmarking process consisted of three steps. First we ported the waveforms to fixed-point implementations, which was necessary since the StrongARM does not having a floating-point unit. Then, we benchmarked the waveforms on a StrongARM evaluation platform to get an estimate of the CPU cycles required, and also on Pentium and PowerPC platforms for comparison. Finally, a subset of the waveforms were actually run and measured on the prototype platform.

The porting process involved removing the floating point code and replacing it with a fixed-point equivalent. This involved two major operations, replacing functions such as arctan or squareroot with look up tables and replacing numbers that were stored as floating point values with a fixed point equivalent, and modifying any computations to insure that the fixed point representation did not overflow.

The set of waveforms that we could run on the prototype were limited due to the frequency and bandwidth limitations of the front-end that was built into the system. The benchmarking procedure is described next, followed by a description of the five waveforms that were benchmarked.

Measurement Procedure

The 5 waveform benchmarks were run on 3 different platforms. The platforms used were:

1) Pentium III (933 MHz) with 256 MB of RAM
2) PowerPC (533 MHz) with 256 MB of RAM
3) StrongARM (200 MHz) with 128 MB of RAM

Fixed and floating point versions of the legacy waveforms were benchmarked on the Pentium III platform and the PowerPC platform. All of the benchmarks are run on simulated waveforms. The waveforms are complex, baseband waveforms at low sampling rates, representing the sample streams that the AD6620 would provide after performing channel selection on a wideband input.

The Sprockit™ source module for each of the benchmarks is a memory source that loads a file (or portion of a file) into a memory buffer during initialization. While operating, the contents of this buffer are copied repeatedly into the output port of the sink. Given the lack of DMA capability on the StrongARM boards, this fairly represents the same operation that a kernel device driver would have to perform to copy the data from the I/O device into the application memory. In each case, the output of the signal processing chains was output to a null device.

To measure the computational requirements of each waveform, a known number of samples at a known sampling rate were processed; this corresponds to a fixed amount of real time. The amount of time actually required to process this input was then measured. The ratio of the two numbers represents the percent of the processing capabilities of the platform required to process that waveform.

Description of Waveforms

1) Legacy FM system

This input waveform simulates the data that would be received from a Motorola TalkAbout radio that uses 25 kHz channels in the 462 and 467 MHz bands. The sample stream is complex, baseband stream sampled at 40 ksps. The signal chain receiving this signal performs the following functions:

1) Amplitude detection (essentially a squelch function)
2) FM demodulation
3) Low-pass filtering of the demodulated signal and decimation to audio rates (8000 Hz).

2) Legacy AM system

This input waveform simulates the data that would be received from a CB radio. The sample stream is a complex, baseband stream sampled at 32 ksps. The signal chain receiving this signal performs the following functions:

1) Amplitude detection (essentially a squelch function)
2) AM demodulation
3) Low-pass filtering of the demodulated signal and decimation to audio rates (8000 Hz).

3) AMPS Forward Control Channel.

This input waveform simulates the data that would be received on the AMPS forward control channel. The sample stream is a complex, baseband stream sampled at 80 ksps. The signal chain receiving this signal performs the following functions:

1) FM Demodulation
2) FSK Demodulation
3) Manchester Decoding
4) Control Codeword Extraction
5) BCH Block Decoding
6) Code-to-text conversion of transmitted messages

4) AMPS Forward Voice Channel.

This input waveform simulates the data that would be received on the AMPS forward voice channel. The sample stream is a complex, baseband stream sampled at 80 ksps. The signal chain receiving this signal performs the following functions:

1) FM Demodulation
2) Low-pass filtering
3) De-emphasis and decimation filtering

Note: this chain does not do the in band data portion of the forward voice channel (which is similar to the forward control channel), and it does contain the expander following the decimating filter.

5) Mobitex Physical Layer.

This input waveform simulates the data that would be received on the Mobitex wireless network. The sample stream is a complex, baseband stream sampled at 80 ksps. The signal chain receiving this signal performs the following functions:

1) FM demodulation
2) FSK demodulation
3) Mobitex Frame Synchronization

4) Mobitex bit descrambling
5) Mobitex block deinterleaving
6) Mobitex block decoding
7) Mobitex interface to data link layer (including CRC)

Note: the signal chain does not include the signal detector following the FM demodulation that distinguishes data bursts from noise.

Results and Discussion

Benchmarking Results

Figure 6 shows the results of the benchmarking experiments. The number in each cell represents the percent of the CPU required to process each waveform in real-time on the given processor. The waveforms are:

- FM: 25 kHz FM
- AM: 10 kHz AM
- AMPS/FCC: AMPS forward control channel
- AMPS/FVC: AMPS forward voice channel
- Mobitex: Physical layer for Mobitex system

These five waveforms were benchmarked on three processors:

- Pentium III with a 933 MHz clock and 256 MB of RAM
- PowerPC with a 533 MHz clock and 256 MB of RAM
- StrongARM with a 200 MHz clock and 128 MB of RAM

Both floating point and fixed point version of the code were benchmarked on the Pentium and Power PC.

Benchmark	PIII/933 (float)	PIII/933 (fixed)	PPC/533 (float)	PPC/533 (fixed)	SA/200 (fixed)
FM	1.6	1.6	2.6	2.0	12.4
AM	0.91	1.0	2.4	1.4	6.4
AMPS/FCC	2.8	2.4	5.1	2.8	21.5
AMPS/FVC	6.5	10.6	11.3	18.6	68.1
Mobitex	8.8	7.2	5.6	3.5	23.5

Figure 6: Waveform Benchmarks (%CPU)

All of these waveforms are capable of running in real time on all of the processors. The StrongARM processor has the added benefit that it's clock speed can be reduced to the level required to just process the data in real-time, resulting in a significant power savings.

Except for the AMPS/FVC, fixed point does as well as or better than floating point on Pentium and PPC. AMPS/FVC is the one benchmark that has a sizeable low pass filter (601 taps). This seems to indicate that the look-up table approach for the more sophisticated floating point functions such as arctan for FM and squareroot for AM improve the computation for the fixed point version, however the scaling required to do a large FIR filter imposes a significant overhead for the fixed-point code.

The PowerPC outperformed the Pentium on the Mobitex benchmark. The Mobitex benchmark made significant use of lookup tables for various functions, and the better performance of the PowerPC is probably a reflection of better memory system performance on that processor. This probably also explains the poor performance of the StrongARM on the Mobitex benchmark, as its memory system runs at a slower speed.

Aside from the Mobitex benchmark, all of the StrongARM numbers seem to be about a factor of 5-7 worse than the PentiumIII. Given the factor of 4.7 in clock speed, not to mention multiple execution pipes in the Pentium, this difference seems to be consistent with the differences in clock speed and architecture.

On the prototype itself we were able to demonstrate a legacy FM system, an analog law enforcement system and a low bit rate data system. The legacy FM system was the Motorola Family Radio Service Walkie-Talkies which operate at 462 MHz, the analog law enforcement system was the analog mode for the Racal MBITR radio, which is an FM system operating at 167 MHz, and the data that consisted of a protocol similar to the AMPS control channel system running in the 400 MHz UHF band.

Discussion

The benchmarking results suggest that implementation of many legacy narrowband systems entirely in software using existing low power processors is feasible today. These results can be used to project the future capabilities of software radio given the expected improvements in low power processors. The processor used in the ITSY prototype was a 200 MHz StrongARM, which dissipates 500 mW while running at the full clock speed. The new version of this processor (called the XScale) runs at 600 MHz while dissipating 450 mW, and dissipates only 10 mW while running at 50 MHz. Future plans for the Xscale call for the processor to reach clock speeds of 1.2 GHz by the end of 2002 while maintaining roughly the same power dissipation at the maximum clock speed.

Second generation (2G) voice standards such as TDMA, GSM and CDMA will require more processing power than the AMPS system. Figure 7 shows preliminary Pentium III benchmarks for AMPS, TDMA and GSM. Benchmarks are preliminary because the software has not been optimized at the time of these measurements. Despite this, these results give us a good relative measure of the processing requirements for TDMA and GSM. With some optimization, it seems reasonable that both TDMA and GSM could run on a 600 MHz XScale processor. CDMA on the other hand, is considerably more computationally complex and will require a more powerful processor.

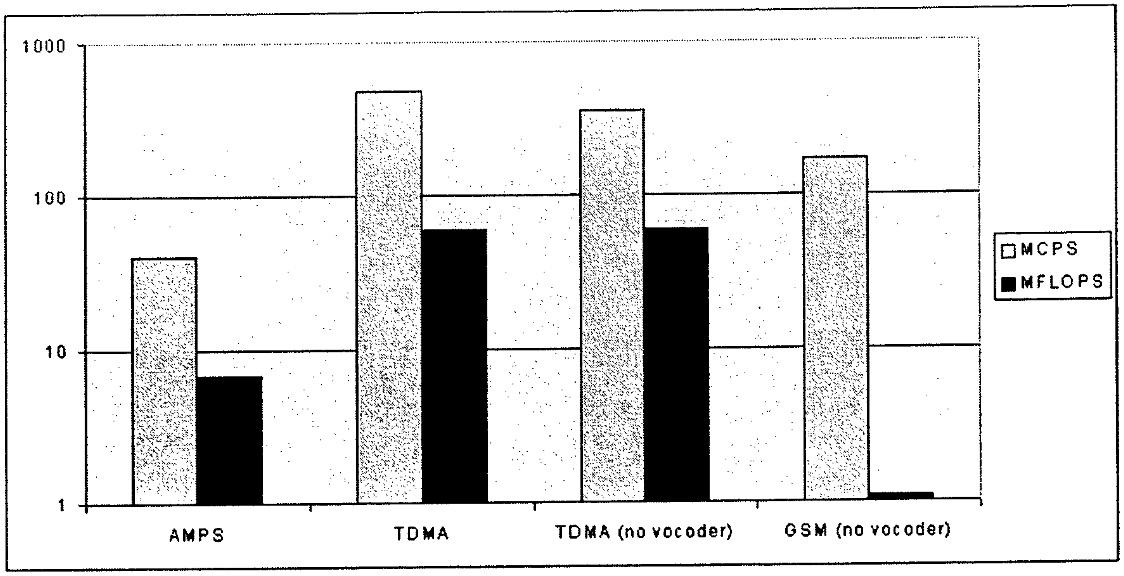

Figure 7: Preliminary Cellular Benchmarks on a Pentium III

Mobitex is a fairly good representation of today's low data rate wireless standards such as CDPD and Ardis. GPRS, an extension of GSM that will provide 115 kbps of wireless data will be a protocol layer addition to GSM, and should only result in a small increase in the required processing power. By contrast, wideband data systems such as 802.11b require significantly more processing. For example the de-spreading function requires more than 200% of a PentiumIII/933, which is more computational power than even the 1.2 GHz StrongARM will have. Software radio is well suited to today's low data rate wireless systems, as well as some of the 2.5G data system that provide roughly 100 kbps data rates, and possibly for a 1 Mbit system like Bluetooth. A pure software radio approach for wideband LAN systems such as 802.11b will not be feasible for the foreseeable future.

Conclusions

This project successfully demonstrated a working handheld software radio. This prototype has served as an excellent proof of concept, and has identified the capabilities and limitations of handheld software radio using existing low power processors. The prototype demonstrated that software waveform processing is feasible for a set of legacy waveforms using existing processors, and most 2G standards should be within the grasp of low power processors such as the XScale that have become available this year. The work also identified wideband networking waveforms, such as 802.11b, as a limitation for the all software approach in a handheld radio, at least for the next few years.

While there is considerable research and development aimed at low power processors for PDAs and cellphones, there is considerably less work on the integrated wideband tunable front-ends required for a multi-band multi-mode software radio. The availability of commercial front-ends limited the scope of demonstration applications that could be run on the prototype.

The major contributions of this work are the demonstration of the feasibility of all-software processing on a general purpose processor for several different legacy waveforms, and identification of the key areas of research and development required to move software radio technology into a viable commercial realm (discussed in the recommendations section at the end of this report), and the measurement of processor performance for five different waveforms that provide a guide for the performance of software radio applications on future processors.

Recommendations

The design and construction of the prototype highlighted several areas in which additional research and development would aid the progress of software radio technology.

Clearly the biggest hurdle was finding a suitable front-end. Our eventual choice of front-end was a compromise of our original goals with what was commercially available. Wide band tunable front-ends do not exist as commodity components, but are technically feasible. Since there are not true multi-band multi-mode radios on the market, there is not driving force behind the development of commercial wideband tunable front-ends. On the other hand, multi-band multi-mode radios are not commercially viable without low cost wideband tunable front-ends. Additional research funding directed at single or multi-chip implementations of RF front-ends may well be the catalyst necessary to solve this chicken and egg problem.

A related area is tunable RF filters, which will be needed to construct high dynamic range, tunable front-ends. MEMS technology looks to be the most promising solution to this problem.

The benchmarks have also provided some guidelines for which functions are the most computationally intensive, and may provide a guide for further research and development of processor instruction sets. Vector instruction sets, such as MMX on the Pentium and AltiVec on the PowerPC have proven to be extremely useful for reducing the computational load of FIR filters and other vector processing instructions. One area where these instructions do not help is in the processing of complex numbers. For example a complex multiplication is actually a cross multiply followed by two additions. Processing of complex numbers is extremely common in signal processing, and instruction set extensions aimed at supporting these functions would be a tremendous benefit.

Finally, the area of low power processing needs considerable additional work. Recent work on clock speed and processor voltage regulation [SC01] is quite promising, but this will need to be coupled with intelligent algorithms [BHM01] and a good operating system abstraction to allow applications to easily take advantage of the various power management functions available on processors and supporting chips today.

References

[BIWG99] V. Bose, M. Ismert, M. Welborn, J. Guttag, "Virtual Radios," *IEEE/JSAC, Special Issue on Software Radios*, April 1999.

[BHM01] V. G. Bose, R. Hu, R. Morris, "Dynamic Physical Layers for Wireless Networks Using Software Radio", International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, UT May 2001

[Chapin01] J. Chapin, V. Lum, S. Muir "Experience in Implementing GSM in RDL (The Vanu Radio Description Language™*), Milcomm 2001*, Vienna, VA, October 28, 2001.

[SC01] A. Sinha and A. Chandrakasan, "Dynamic Voltage Scheduling Using Adaptive Filtering of Workload Traces", *Proceedings of the 14th International Conference on VLSI Design*, Bangalore, India, January 2001.

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.